IN THE CLAIMS

1.-10. (Cancelled).

11. (Currently Amended) A method comprising:

dividing a dynamic sequential program into multiple epochs comprising a first epoch instance and a second epoch instance, wherein each epoch includes [[2]] two or more instructions;

in a redundant multi-threading (RMT) system having leading and trailing threads, redundantly executing in parallel first epoch instance and second epoch instance ~~instances for each epoch~~ as the leading and trailing threads, respectively, in the RMT system;

for the executed first epoch instance and second epoch instance ~~instances~~, saving ~~epoch instances~~ store results of the first epoch instance and the second epoch instance as speculative stores to memory, the speculative stores being exposed;

comparing the exposed stores; and

if [[they]] the exposed stores match, committing ~~a single set of~~ one of the exposed stores atomically to a sequential architectural memory state of a computation corresponding to the dynamic sequential program.

12. (Previously Presented) The method of claim 11, wherein the speculative stores are from a re-order buffer.

13. (Currently Amended) The method of claim 12, wherein the two or more instructions executed in response to the execution of the first or second epoch instances are buffered prior to epoch execution completion.

14. (Cancelled).

15. (Previously Presented) The method of claim 11, wherein the memory is Ll cache memory.

16. (Currently Amended) A method, comprising:

breaking a program to be executed into multiple epochs each having two or more instructions;

redundantly executing the program by redundantly executing each epoch separately, and

sending speculative results for each epoch to memory;

checking the speculative results for each epoch against each other; and

if they match, committing the results atomically to a sequential architectural memory state of a computation corresponding to the program.

17. (Previously Presented) The method of claim 16, in which the speculative results are speculative stores.

18. (Previously Presented) The method of claim 16, in which the memory is L1 cache memory.